THE FIRE EFFECTS INFORMATION SYSTEM : AN ASSESSMENT

BY
LARRY S. BRADSHAW

FINAL REPORT FOR COOPERATIVE AGREEMENT with SYSTEMS for ENVIRONMENTAL MANAGEMENT
#INT-89393

FS Contact: C. M. Johnston (ADOR)
Cooperator Contact: Collen D. Bevins (PI)

SYSTEMS FOR ENVIRONMENTAL MANAGEMENT

P.O. Box 8868
Missoula, MT 59807
Phone: (406) 549-7478

The Fire Effects Information System :

An Assessment

Final Report

Larry S. Bradshaw and Collin D. Bevins
Systems for Environmental Management
Missoula, MT 59806

1. INTRODUCTION

The purpose of Research Joint Venture Agreement INT-89393-RJVA is to review and evaluate the current Fire Effects Information System (FEIS) and make recommendations for future development. The specific objectives are:

1. Evaluate alternative programming languages for the Fire Effects Information System.
2. Recommend programming improvements for the Fire Effects Information System.

To meet the objectives, the following tasks were performed:

1. A study plan was prepared and accepted in collaboration with the Forest Service.
2. All available FEIS documentation was reviewed to gain an understanding of its form and function.
3. The FEIS Builder and Query programs were used with a sample sagebrush textbase.
4. The structure of the sample sagebrush text file was closely examined.
5. Interviews were conducted with biologists who currently use FEIS to enter, edit, and maintain text information.
6. Under the considerable advantage of evaluating FEIS after its form and function have become well defined, the feasibility, ease of use, and appropriateness of alternative text storage and retrieval paradigms were examined.

In this report we present the following topics:

1. Our perception of the current FEIS based upon our readings about, and use of, the system and interviews with those who coexist daily with FEIS.
2. Our perception of FEIS attributes which encumber, rather than enhance, its use and functionality.
3. An operational summary of the FEIS subsystems.
4. A discussion of three alternatives for implementing FEIS.
5. An estimation of costs for each alternative.
6. Our recommendation for further development.

2. THE CURRENT STATE OF FEIS

FEIS Attributes

Primary characteristics of the current FEIS system are summarized below.

- FEIS is a topic-based text management system, written in COMMON LISP, containing two subsystems, the Builder and the Query.
- FEIS text information is stored in a single file and referenced by a single index file. The text file currently contains over 7 million characters of information on 202 vegetative species, 11 cover types, and 10 wildlife species.
- Topics are arranged in a simple hierarchical relationship with plant species, wildlife species, and ecosystem at the root.
- FEIS segregates text from the hierarchical structure into frames and slots and stores them as independent blocks within the single text file.
- The hierarchical relationship between frames and slots is maintained internally by a frame index and frame/slot pointers.
- The Builder is used to modify (add/delete/edit) text within the primary textbase. The Builder program maintains the text index file.
- Only one person at a time may use the Builder program on the primary textbase.
- Builder access to textbase information is limited to a single slot at a time.
- A duplicate Builder, the Alt-Builder, is used by biologists to enter and modify text in species-specific edit-files.
- A Merge utility combines edit-files into the primary textbase.
- Builders add text to empty slots by menuing to the desired slot and (1) reading a text file into the selected slot, or (2) calling a text editor (EMACS).
- Builders modify existing slots by (1) menuing to the desired slot, (2) extracting the text slot from the textbase to an edit file, (3) editing (EMACS) the edit file, and (4) appending the edit file to the textbase.

- A Reorganize utility periodically packs the textbase file to remove dead space created during the edit cycle.

- Query locates, collects, and displays species information from the primary textbase using a structured menu interface. Query users are not allowed to modify the text.

- Query is extremely easy to use and requires little or no instruction or computer experience to navigate the menus and view information.

- Query can save all information on a species to a disk file, which may later be printed or mailed to a remote user.

- Query does not allow ad-hoc word or phrase searches.

- Query does not allow printing of selected blocks of text.

- Query does not interface with the FEIS companion bibliographic reference system containing all FEIS literature citations.

- A separate utility program, Extract_Data, can extract species information into new textbases on the basis of species, ecosystem, or administrative unit. However, there apparently are problems with this program as it works only on a very limited basis.

FEIS Limitations

The above attributes that most encumber FEIS are:

- FEIS does not provide several basic capabilities of a text management system; ad-hoc phrase and keyword searching, printing of selected text blocks, and ease of maintenance.

- FEIS programs on the Data General are large. About 7 megabytes of computer memory are required to load and run either the Builder or Query programs.

- The frame/slot data structure requires an unnecessarily complex system of linked-list pointers to keep track of simple sequential text records. The complexity of the text structure is propagated through the software, which must understand how to access and collate the text.

- Builder access to the data structure forces an inefficient edit cycle. Slots must be accessed, extracted, edited, and uploaded, **one at a time** using the Builder programs. This prevents use of global edit (search, search & replace) functions across any block of text larger than a slot. As a result, information entry and maintenance within the FEIS environment is cumbersome and inefficient.

3. OPERATIONAL SUMMARY

Query Subsytem

The Query subsystem is designed for the casual user with little or no computer training or experience. In that capacity it performs exceedingly well. Using the PC version of the Query program (FIRESYS) we had no trouble accessing, traversing, and viewing our sample database. We did note that the SAVE option on the PC version caused a non-recoverable program error.

The menuing scheme contains a few inconsistencies that should be removed. In particular FEIS should allow:

- o process termination without having to first use the Options menu, and
- o users viewing all information on a particular species to do so without traversing all the frame access menus.

Builder Subsytem

Program Design and Implementation

The Builder subsystem is designed for users that are expected to be knowledgeable about computers. However, even for the biologists who have been working with FEIS for several years, the piecemeal approach of maintaining the textbase is often a frustratingly slow process.

Builder program size and implementation compound the problems associated with the system design. The Builder programs reside on an MV/4000 computer with 8 megabytes of main memory which is accessed as a 'deflected drawer' from a MV/10000 computer. At 7 megabytes per program image, performance rapidly degrades when one person is using the Builder and one or more persons are using the Alt-Builder. Further degradation occurs as non-FEIS users go about their daily business of compiling FORTRAN programs, doing statistical analysis (SAS), etc. on the MV/4000.

Text Entry

All word processing is done on the MV/10000, and MV/10000 files are not directly accessible from the MV/4000. It takes six steps to fill a FEIS text slot (of which there 20-30 per species):

1. prepare document on the CEO side of the MV/10000,
2. export document to the AOS/VS side of the MV/10000,
3. deflect to the MV/4000,
4. retrieve file from the MV/10000 using the retrieve utility,
5. use the Builder program menus to access a slot, and
6. read the file into the slot.

Alternatively, the EMACS editor is used to enter text into a slot. This has the advantage of skipping the transfer steps (which can be confusing and are un-documented), and the disadvantage of using EMACS as the main text entry tool. As an editor EMACS, is adequate for the job; as used it's the bane of the biologists work day. There are four main reasons:

- The DG EMACS users manual is intimidating and difficult to comprehend, resulting in an aversion in learning to take advantage of EMACS capabilities.

- EMACS on the MV/4000 is slow at echoing what is typed in at the keyboard. There is a time delay between when a key is pressed and when it appears on the screen. Cumulatively, an average typist can get several paragraphs ahead of what is on the screen. The delay increases as multiple users use the Builder programs, and is more irritating to those typing in prepared text than to those composing at the computer.

- EMACS is invoked as a 'son process' from within FEIS, further adding to MV/4000 memory allocation demand.

- A lack of training in using EMACS for more than the most basic editing skills required for the job.

Copy Editing

A part-time editor edits text for grammar, spelling, style and format. Generally a write-up is not edited until **after** it has been merged into the primary textbase. The implications of this are:

- The editor must learn the data (frame/slot) relationships,

- The editor must use the Builder/EMACS interface to edit the primary textbase, slot by slot.

Text Display Windows

The Builder programs displays text on screen rows 8 to 21. The Query program displays text on screen rows 7 to 21. This sometimes results in inconsistencies between what a Builder user and a Query user sees while viewing the same slot. Formatted tables are particularly susceptible to this problem, as is the copy editor, who must edit from a display window different than the Query.

Documentation and Training

There is not a standard training resource for new biologists working with FEIS. Training is basically show me and word of mouth, which is detrimental to both the new, and the working biologist. There should be a training manual that contains:

- Job description and expectations.
- Documentation of tasks from species assignment to merge.
- Brief, usable instructions for

  1. using FEIS subsystems,
  2. using CEO (particularly word processing),
  3. exporting CEO files to AOS/VS
  4. retrieving files from the MV/10000, and
  5. using the EMACS text editor effectively.

Utility Programs

We did not have access to the Utility programs:

Initialize index,
Reorganize data,
Check data,
Extract data,
Create data,
Merge data,
Print data dictionary, and
Compile source files

and cannot comment on their function or performance.

ALTERNATIVE 1: KEEP FEIS IN LISP

If FEIS is kept in LISP with the current data structure, the following actions should be considered.

- Modify Builder programs so that a complete species write-up can be edited at once. This change would allow:
  - more efficient text entry,
  - more efficient text maintenance, and
  - use of global edit functions on a frame level.
- Modify Builder program menuing schemes to allow more flexibility.
- Modify Builder programs so users can view previous pages and terminate the viewing process on demand.
- Edit species before they are merged into the primary textbase.
- Make the Builder and Query text display windows the same.
- Investigate alternative text editors that are faster and easier to use.
- Develop a training package to help Builder users take full advantage of the text editor (EMACS).
- Develop an instruction manual for each of the alternative processes of entering and maintaining text on the system.

ALTERNATIVE 2: FEIS USING ORACLE

ORACLE is a popular relational data base manager (RDBM) featuring several 4th generation front end tools. It is available on a wide variety of computers and operating systems including Data General AOS/VS and IBM PC-DOS. It has been prescribed by the Forest Service as its DBMS of choice.

The strength of traditional RDBM's lies in the storage and ad hoc access of numeric and character-based data. The fact that they are not optimized for character-only data is reflected in the growing popularity of text data base managers (TDBM) such as AskSAM.

RDBMs store their information in tables. Each table contains one or more rows, and each row contains one or more columns. Each row must be uniquely identified by one or more of its columns; this column combination is called a key.

Storage of standard formatted text inside an RDBM requires a key for each line of text. The key must not only identify the species and its hierarchical subject list, but also a line sequence number. The length of the key, repeated for every line of text, significantly increases the size of the database. Furthermore, the keys are also stored in an index, using even more storage space.

RDBM's do not typically include usual text editing functions such as block moves, word wrap, and line insertion/deletion. To insert or delete lines using RDBM tools, the user takes responsibility for creating new, unique keys, including line numbers. To insert or delete characters within a line, the user assumes responsibility for performing word wrap, which often necessitates editing several subsequent lines (table rows).

ORACLE is not well suited to the tasks required of FEIS. Its use would increase file storage requirements, increase the number of software components to be mastered by both users and maintenance personnel, decrease ease of use, and increase the edit cycle.

- The user would first enter or edit text using a word processor or text editor (ORACLE's SQL*FORMS does not have traditional word processor functionality such as word wrap).
- A program must be available and run against the text file to convert it into tabular format.

- The text table would include certain keys repeated for each line of text. The minimum required keys would be a species identifier, a subject identifier, and a line number. These keys are required just to keep the text in sequence and extract it by species and subject; it does not enable keyword searching.

- The keys are repeated on each line of text, significantly increasing file size.

- The table file must be loaded into ORACLE using SQL*LOADER.

- ORACLE must allocate space for not just the table data, but also the key index.

- Inserting, editing, or deleting multiple text lines within ORACLE would be very difficult. Not only are there are no block move or word wrap facilities, but the user would be entirely responsible for generating and maintaining text line keys.

- To modify text, the user would have to SELECT the text from ORACLE to an ASCII table file, run a program to convert the table file to text format, edit the file, convert the text file back to table format, then load it back into ORACLE. The inefficiencies of the process exceed those of the current Builder program.

- The user interface for querying and scrolling text information would be vastly different from the current Query program. The current FEIS appeal to casual users would be lost.

- Since it is not the intent of FEIS to provide ad hoc conditional or relational keyword searches, the user pays all the costs of using ORACLE (large overhead), and realizes none of the benefits.

- Since FEIS does not currently interface with other Forest Service data structures, there is no additional benefit to using ORACLE in this respect.

ORACLE may be an appropriate tool if all the following circumstances were present:

- FEIS employs a well-defined set of keywords to perform conditional keyword searches (as do many citation retrieval systems).

- FEIS somehow interacts with other Forest Service corporate data structures residing in ORACLE (such as timber inventories)

- FEIS end-users are trained to perform queries using ORACLE's 4GL tools.

ALTERNATIVE 3: NEW FEIS STRUCTURES, PROGRAMS, AND LANGUAGE

The following describes an alternative text management system based upon our current knowledge of how FEIS is maintained and used. The system is comprised of new text file structures and text processing software designed to simplify and enhance FEIS.

Text Structures

The proposed system consists of four types of ASCII text files; one master species list file, one keyword file, one header file, and many species text files.

Plant, wildlife, and ecosystem text is stored in final display format in separate ASCII text files for each species. The merits of this organization are:

- All information about a plant, animal, or ecosystem is stored in one place.
- Biologists maintaining the system are able to work directly with all the information relevant to a species in a what-you-see-is-what-you-get environment.
- To create new text files, a skeleton plant, wildlife, and ecosystem file is available containing all the subject headings. The biologist uses the editor to insert text in the appropriate place.
- To edit a text file, the biologist simply accesses the file from his/her favorite word processor or text editor. All the block move, search and replace, and other editing functions are available at one time and place on all text related to the species.
- 'Builder' software is not needed to access and maintain frames and pointers. To view, print, or edit species information, maintenance personnel need only TYPE, PRINT or edit it.
- Responsibility for allocating, packing, and freeing file space is left to the operating system where it belongs.
- The display (Query) program does not need to traverse pointer lists to accumulate text information.
- This type of organization is easily understood, making it easy to access, maintain, and extend FEIS capabilities.

The master species list is maintained to identify all plants, animals, and ecosystems in FEIS.

- The file contains the species name, abbreviation, text file name, and any other information (e.g. life form) to appear on the display program's main species menu.
- The file contains additional housekeeping information such as species text file dates, times, and generation to verify file currency.
- The file is maintained by a utility program (lets call it UPDATE) that reads an edited species file, checks it for completeness, and updates its entries in the master - species list.
- The file may be accessed by report utilities to produce indices, tables of content, and summaries of FEIS size, content, and currency.
- The file is in tabular ASCII format to simplify access, viewing, printing, and maintenance.

The key file is used to rapidly locate and access species text files based upon selected key values.

- The key file is a simple ASCII text file containing key-value-file triplets.
- The key field is a character string identifying the type of key. Under current FEIS usage there are three keys; STATE, ADMINISTRATION, and ECOSYSTEM.
- The value field is a character string identifying the value of the key. Valid values for STATE include MT and ID.
- The file field is a character string identifying the plant, animal, or ecosystem text file containing the key-value.
- Because of its simple structure, keys, values, and files may be defined, inserted, updated, or deleted as the need arises.
- The key file is maintained by the UPDATE program.

The header file associates a display header with each topic header in the species text files.

- The header file is a simple ASCII text file containing level-tag-header triplets.
- The level field is a character string identifying the subject's hierarchical position in the text. The string '010203' would indicate primary level 1, secondary level 2, and tertiary level 3.
- The tag field is a character string identifying the subject's tag in the species text file. Tag fields begin with a special character, such as '@'. The tag for species name, for example, would be '@SPECIES-NAME'.
- The header field is the character string that replaces the tag during display. For the above example, the header string could be 'Species Name'.
- A text editor is used to insert, edit, and delete display headers on an as-needed basis.

Software

The proposed system requires only 2 computer programs.

The UPDATE program performs the following tasks:

- reads the specified (newly created or edited) species text file,
- checks its contents for completeness against the header file,
- extracts keys and updates the key file, and
- updates the master species list file with the species name, abbreviation, and file name, date, and time.

The DISPLAY program performs the following tasks:

- reads the master species list file,
- constructs and presents a species menu,
- reads a selected species text file into memory,
- tests the species file's currency,
- dynamically builds a subject menu and index while reading the file,
- allows the user to scroll or page up or down through the species text,
- allows the user to pop up a subject menu, select a topic, and jump to its position in the text file,
- allows the user to search for any arbitrary character string within the species file,
- allows the user to print arbitrary blocks of text to a disk file.

Many other programs could be written to support FEIS. Some suggestions are:

TOC	Reads the master species list file and produces a Table of Contents summarizing the current state of FEIS.
CHECK	Similar to UPDATE, but only tests the species text file for completeness.
CURRENCY	Compares species text file dates and times with the time stamps in the master species list. Ensures that all edited species text files have been UPDATEd.
KEYSEARCH	Reads a list of key-value pairs, searches the key file, and produces a list of all text files referencing them.
KEYCOPY	Reads a list of key-value pairs, searches the key file, copies all text files referencing them into another directory, and creates a new master species list file, key file, and header file in the directory.

Programming Language

The proposed system would be written in ANSI 77 FORTRAN. The language is highly standardized, portable across FEIS targeted computers, supported by the Forest Service, and well understood by Forest Service programmers. FORTRAN terminal IO libraries such as F77SCREENS and FSAT are available for the Data General, and can be emulated on PCs. FORTRAN's string management facilities are adequate for the outlined tasks.

The combination of FORTRAN 77 with standard ASCII text files yields the maximum assurance of portability across many computers and operating systems.

COST COMPARISONS

Program Development Costs

LISP — $??,??? committed to date for FEIS software development to maintain linked list data structures using LISP.
$??,??? needed to make recommended program changes and fixes.

ORACLE — $ 40,000 projected to manipulate text files, table files, and database tables using ORACLE tools and SQL embedded in FORTRAN.

FORTRAN — $ 25,000 projected to manipulate ASCII text files using FORTRAN. (Includes utilities TOC, CHECK, CURRENCY, KEYSEARCH, KEYCOPY)

Program Maintenance Costs

LISP — Projected to be high because (1) LISP is not widely understood in FS, (2) the underlying linked list data structure is complex, and (3) the initial development costs (which reflect maintenance costs) were high.

ORACLE — Projected to be high because two software systems, FORTRAN and ORACLE, must be understood, documented, coordinated, and maintained.

FORTRAN — Projected to be moderate because a single language is used to maintain standard ASCII text files.

Text Database Development and Maintenance Costs

Estimating an accurate cost/species value is difficult. Two biologists, Nanka McMurray and Debra Tirmenstein have been involved basically full time since summer, 1986. Chuck Bushey did some wildlife write-ups in 1986. Each summer several additional people have worked full time on the project, and each school year several students work part time. Outside a core group of Nanka, Debra, and Dorothy (copy editor) there has been a rather high turnover rate of people. Last year with the addition of Ron Uchytil, and the recent addition of Marti Crane (both full time), the working group has stabilized at about five people. Four staff people are involved in various ways. Bill Fischer and Anne Bradley manage the project and review species write-ups. Anne, Cam Johnston, and Elizabeth Reinhardt spend various amount of time trouble shooting and modifying the programs and maintaining the main database (merges, reorganizations, etc.).

We estimated that, aggregated, four persons have worked full time since summer 1986. We compared this to 240 species in the system. (There are currently 233 in FEIS). This comes to 12 person/years (20 species/year/person). Using pay periods as a base, we estimate 1.3 pay periods per species. At 10 days/pay period, this comes to 13

days/species.

More important than the actual number of days per species is the percentage of time the biologists could save by working in an efficient environment. We estimated that with:

1. a data structure that allows global editing of an entire species,
2. a word processing type editor instead of EMACS,
3. removal of the cumbersome BUILDER program and menu structure, and
4. simplified training and lower turnover of personnel,

biologists and the copy editor could reduce their computer interaction time by almost 50 percent per species. This would result in a large reduction in resource demands on the computer, and also greatly improve the working environment. There are few things more frustrating than working in a rigid environment on a sluggish computer, even when paid by the hour.

Summary - Text Database Development and Maintenance Costs

LISP	Experienced at approximately 13 person/days per species. Basis: 4 full time persons for 3 years and 240 species in FEIS. May be reduced to 10 days/species with recommended changes.
ORACLE	Projected at 18 person/days per species. Basis: Increased by downlowding/uploading data.
FORTRAN	Projected at 8 person/days per species. Basis: 50 percent reduction in computer time.

Text Database Conversion Costs

LISP	$ 0
ORACLE	$ 15,000
FORTRAN	$ 10,000

ALTERNATIVE SUMMARY

	Alternative 1 LISP (Modified)	Alternative 2 ORACLE	Alternative 3 FORTRAN
Program Development	$??,???	$ 40,000	$ <25,000
Program Maintenance	High	High	Moderate
Program Size	Large (7 MB)	Small (< 1 MB)	Small (<1 MB)
Database Development	10-12 days/spp	18 days/spp	8 days/spp
Database Conversion	$ 0	$ 15,000	$ <10,000

RECOMMENDATION AND RATIONALE

RECOMMENDATION:

1. Rewrite FEIS in FORTRAN (as specified in Alternative 3), preserving the integrity of the text base, ensuring the new QUERY program (DISPLAY) program operates in much the same manner as the current QUERY but with expanded ad-hoc search and print features.

2. Develop a FEIS instruction manual to teach FEIS use, structure, and maintenance procedures. The manual should explain helpful and commonly used operating system and editor features.

RATIONALE:

1. The evolved state of FEIS does not require Artificial Intelligence techniques (programs or data structures) initially used during the development and protoyping of the system.

2. Conversion to FORTRAN will (1) ensure lower program maintenance and modification costs, (2) significantly reduce code size, and (3) increase system portability. These cost savings will be realized whether FEIS is maintained in-service or out.

3. Use of ASCII text files (1) reduces program complexity, (2) reduces maintenance costs, (3) improves system expandability, and (4) removes the need of the BUILDER programs. The BUILDER programs, the complex data structure, and the piecemeal way the two interact account for the majority of the problems in building and maintaining the FEIS textbase.

4. Removal of the BUILDER/EMACS edit cycle will dramatically increase text base development and maintenance efficiency.

SOURCES

System Documentation, FIRESYS - Fire Effects Information System.
Alden W. Wright and others. University of Montana, Dept. of Computer Science. Missoula, MT. June 1987.

FIRESYS - Using artificial intelligence techniques to build a fire effects information system.
William C. Fischer and Alden Wright. The Compiler, Vol 5. No. 5 1987.

Instructions for FEIS species write ups. W.C. Fischer. 1987. On file, IFSL.

EMACS Users Manual for the Data General. Data General Corporation.

Understanding Oracle. Sybex Inc. Alameda, CA. 1989.

FIRESYS Data Dictionary.

Interviews with:

Anne Bradley
Nanka McMurray
Ron Uchytil
Debra Tirmenstein
Marti Crane
Cameron Johnston